REMARKS

Claims 1, 4-10, 12-16 and 19-22 were pending at the time of examination. Claims 1, 4-5, 10, 13, 16, 19 and 21 have been amended. No new matter has been added. The applicants respectfully request reconsideration based on the foregoing amendments and these remarks.

Claim Rejections – 35 U.S.C. § 103

Claims 1, 4-8, 10, 12, 14, 16, 19, 21 and 22 were rejected under 35 U.S.C § 103(a) as being unpatentable over U.S. Patent No. 5,991,173 to Unger et al. (hereinafter "Unger") in view U.S. Patent No. 6,163,811 to Porter et al., and in further view of Ainon "Storing text using integer codes," 1986, *Proceedings of the 11th conference on Computational linguistics* (hereinafter Ainon). The applicants respectfully traverse this rejection.

Claim 1 has been amended by incorporating limitations similar to the limitations of original claim 4, as well as further limitations, to specifically state that the method refers to a method of compiling a computer source program into a compressed compiler product containing differential names.

Claim 1, as amended, recites:

> "receiving a source program written in a high-level programming language, the source program including one or more program symbols and non-program symbol information;
>
> encoding a program symbol name to produce an encoded program symbol name, without changing the non-program symbol information;
>
> determining a differential name for the encoded program symbol name relative to a base symbol identifying a containing scope for the program symbol, wherein the containing scope is selected from a group consisting of: a namespace, a package, a module, a container object, and a function, and defines a context within which the differential name has an unambiguous meaning, and wherein the differential name is formed at least in part by a sequence of characters constituting a subset of the encoded program symbol name; and
>
> replacing the encoded program symbol name with the differential name to facilitate producing a compressed compiler product including the generated differential name."

All of the Unger, Porter, and Ainon references have been discussed in detail in the previous office action responses, and will therefore only be addressed herein with regards to the specific amendments made in this response. Generally, Unger and Porter describe various compression methods for data structures. Unger describes methods for compressing natural language text, using a set of dictionaries, while Porter describes token-based source file compression. Ainon describes a coding method of text, in which a word list with syntactic linear ordering is stored and words in a text are given two-byte integer codes that point to their respective positions in this word list. However, neither Unger, Porter, nor Ainon reasonably suggest or describe claim as presented.

In section (A) of the "Response to Arguments" section in the most recent Office Action, the Examiner argues in essence that since claim 1 recites determining a differential name for the encoded program symbol name, the determination is made for a sequence of alpha-numerical atomic elements, and would thus be rendered obvious by Unger's text compression method. The applicants respectfully disagree. It is true that the encoded program symbol names are sequences of alpha-numerical elements. However, as has been discussed before, program symbol names are not simply characters in a text file as taught by Unger, but are more correctly characterized as programming references or pointers with meaning outside of the contextual of a text file. The meaning of the program symbols is maintained when the program symbols are converted into encoded program symbols. In contrast, Unger's method of compression compares the words (including *numeric strings, decimal points, currency symbols*, etc.) to a predetermined dictionary, and then to a supplemental dictionary if the words are not found in the previous (*see* col. 10, ll. 23-39). Numeric characters can be encoded with a special predetermined "numeric" dictionary (*see id.*). As such, the applicants maintain that the symbols described in Unger are merely characters in a text file that have no function or meaning outside of the context of the text file being compressed, and thus that they cannot be equated with the encoded program symbols of the applicants' invention. Furthermore, in Unger, the goal is to encode as much as possible of the text file, while in the applicants' invention, only the program symbol names are encoded and the non-program symbol information is left unaltered, as specified by claim 1.

In section (B) of the "Response to Arguments" section in the most recent Office Action, the Examiner argues that the combination of Unger, Porter and Ainon discloses a reduced-size format of a differential, and that claim 1 is not specific enough in terms of the "reduced-size format" and the "containing scope" to distinguish it from the combination of references. The applicants have amended claim 1 to be more specific with respect to these issues. As can be seen in the "determining..." step, the containing scope has been further specified in that it "defines a context within which the differential name has an unambiguous meaning." That is, the differential names may have different meanings depending on the "environment" in which the differential names operate. However, as soon as the differential name is interpreted with respect to its containing scope, there is a clear an unambiguous meaning of the differential name. Claim 1 further specifies examples of such containing scopes to be "selected from a group consisting of a namespace, a package, a module, a container object, and a function." It is respectfully submitted that the definition of "containing scope" in claim 1 is now completely clear, and that such a containing scope is neither disclosed nor suggested in Unger, Porter or Ainon, taken alone or in combination. With respect to the "reduced-size format" limitation for the differential name, this limitation has been replaced with a more explicit limitation specifying that "the differential name is formed at least in part by a sequence of characters constituting a subset of the encoded program symbol name." This limitation specifies more clearly how the differential name is formed, and since it is formed from a subset of characters of the encoded program symbol name, it is inherent that the differential name has a reduced size compared to the corresponding encoded program symbol name. It is respectfully submitted that the modified definition of "differential name" in claim 1 with respect to its size, is now completely clear, and that such a differential name is neither disclosed nor suggested in Unger, Porter or Ainon, taken alone or in combination.

The last step of claim 1, as amended, requires "replacing the encoded program symbol name with the differential name to facilitate producing a compressed compiler product including the generated differential name." In both Unger and Porter, the words/symbols (which again are different from encoded program symbol names) are replaced with tokens whose meaning is defined elsewhere. Similarly, in Ainon, the words are replaced by two-byte integer codes that represent positions in word lists where the respective words are stored. Neither the tokens, nor the two-byte integer codes are differential names formed by sequences of characters constituting subsets of encoded program symbol names, as required by claim 1, and whose exact meaning can be deduced with reference to the containing scope.

Generally, it should also be noted that, as stated in the specification of the application,

> "the encoded symbol names are typically substantially longer than the original context-dependent names. As can be seen in Figs. 1 and 2, even in the simplest cases, the encoding may result in encoded identifiers that are ten times longer than the original context-dependent identifier. Moreover, in more practical applications, symbols can have many levels of containers, with many containers having very complex names. In such cases, the length of the encoded identifiers for symbols can be very long. Encoded identifiers in excess of 5000 characters have been reported in some applications."

Thus, one skilled in the art would appreciate that encoded context-dependent identifiers (encoded program symbol names) are often difficult to compress because of their context and uniqueness. Generally speaking, compression schemas depend on repeated characters alone and in combination. Thus, conventional compression techniques (like those described in Unger, Porter and Ainon) often leave encoded program symbol names unaltered and uncompressed, in order to maintain a reasonable number of tokens or integer codes. As such, neither Unger, Porter, nor Ainon, alone or in combination, render claim 1 obvious. Again, Unger does not describe compression of encoded program symbol names. Porter does not cure the deficiency in Unger because Porter describes only compression of entire source files over a distributed network. Furthermore, while Porter describes creating a symbol table for operands (i.e. "=", "+", "-", etc.) and their substitution (*see* col. 6, ll. 17-27; FIG. 3a), Porter does not describe encoded program symbol names or differential names as contemplated by the present claim. Ainon adds nothing, rendering claim 1 more obvious than the combination of Unger and Porter alone, since it merely discloses an alternative compression mechanism. For at least the above reasons, it is respectfully submitted that the rejection of claim 1 is unsupported by the cited art and should be withdrawn.

Claim 16 is a *Beauregard* claim corresponding to claim 1. For reasons substantially similar to those set forth above, the applicants respectfully contend that the rejection of claim 16 is unsupported by the cited art and should be withdrawn.

Claims 4-9 depend from claim 1, and the rejections of these claims are unsupported by the cited art for at least the reasons discussed above with regards to claim 1, and should be withdrawn.

Claims 19-20 depend from claim 16, and the rejections of these claims are unsupported by the cited art for at least the reasons discussed above with regards to claim 16, and should be withdrawn.

Claim 10 describes a method for generating encoded program symbol names in an uncompressed form, and was rejected for the same rationale that was set forth in the rejection of claim 1. Claim 10 contains limitations relating to program symbol names, base symbols, and differential program symbol names and formats. Consequently, for at least the reasons discussed above with regards to claim 1, the applicants respectfully contend that the rejection of claim 10 is unsupported by the cited art and should be withdrawn.

Claim 12 depends from claim 10, and the rejection of this claim is therefore unsupported by the cited art for at least the same reasons, and should be withdrawn.

Claim 21 is a *Beauregard* claim corresponding to claim 1. For reasons substantially similar to those set forth above, the applicants respectfully contend that the rejection of claim 16 is unsupported by the cited art and should be withdrawn.

Claim 13 was rejected for substantially the same reasons as claim 1. Claim 13 includes the limitation that the enhanced compiler includes "one or more differential names corresponding to the program symbol names." The program symbol names and the differential names have been discussed above with respect to the rejection of claim 1. For reasons substantially similar to those set forth above with regards to claim 1, the applicants respectfully contend that the rejection of claim 13 is unsupported by the cited art and should be withdrawn.

Claims 14 and 15 depend from claim 13, and the rejections of these claims are unsupported by the cited art for at least the reasons discussed with regards to claim 13, and should be withdrawn.

Conclusion

The applicants believe that all pending claims are allowable and respectfully request a Notice of Allowance for this application from the Examiner. Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

[illegible]

Fredrik Mollborn
Reg. No. 48,587

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300